Statistical Techniques in Robotics (16-831, S20) Lecture #3 (Wednesday, February 8)

Halving Algorithm, Randomized Greedy Algorithm, Regret

Lecturer: Kris Kitani *Scribes: Zhengyi Luo, Erica Weng*

1 Recap

In the previous lecture, we discussed one algorithm for the problem of Prediction With Expert Advice (PWEA) – the Consistent Algorithm. PWEA is a problem of online learning where the system takes decisions based on advice from a set of experts over time with the goal to be as good as possible as the best expert in the group, if not better. Over time, the system learns from making mistakes based on the advice from each expert and updates the hypothesis function which modifies the weighting given to each expert in the group. The PWEA problem is one-shot, instructive, and exhaustive. The different ways to choose your hypothesis function and weight update strategy give us different variations of the Consistent Algorithm.[1, 2]

2 Online Learning

We formally define online learning and differentiate online learning from statistical learning (e.g., supervised learning). Specifically, statistical learning assumes that the input data consist of i.i.d. (independent and identically distributed) random variables. Namely, each input data/random variable has the same probability distribution as the others and are all mutually independent. In online learning, we make no such assumption, and the input data comes from a stream and the distribution of the data can change over time. The data samples are not identically distributed and can even be generated by an adversarial agent.

2.1 Adversarial agent

In online learning, the input sequential data can be generated by an "adversarial agent" or "nature" that is working against our algorithm. For instance, assume a game of questions answering where the learner's objective is to learn from a sequence of questions and answers provided by the host. The host can deliberately design questions that have conflicting answers such that the learner can never learn from his/her mistakes and can never win. It is possible that with proper design the questions can never be answered perfectly. On the other hand, if we assume realizability, then there exists a perfect mapping from the question to the correct answer.

3 Predicting with Expert Advice Problem Setup

In this lecture, we continue to study the online learning problem called "Predicting with Expert Advice Problem Setup". The problem is identical to that used in the last lecture. Concisely, we have: the instance domain $\mathcal{X}$, or the set of possible observations that can be seen at any timestep t, which is the information/advice from experts provided to us for making a decision. We have the target domain $\mathcal{Y}$, also known as the output space, which is the set of possible outcomes that can occur. We have a hypothesis class $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$, a series of functions mapping an element in the instance domain to an element in the target domain, each of which determines how we will make a prediction based on the observations we see. The Version space $\mathbf{V}_t$ at time t is defined as the set of experts whom we still trust at time t; all experts in V_t have not yet made a mistake at time t. We assume realizability (learner has access to the perfect hypothesis) for both algorithms described in these notes.

4 Halving Algorithm

The Halving algorithm improves the mistake bound of the Consistent Algorithm significantly. At each timestep t, instead of picking the first expert from the pool of experts who have been consistently correct in their predictions until t, we will pick the decision which has at least $\frac{1}{|\mathcal{Y}|}$ expert representation ($|\mathcal{Y}| = 2$ in the PWEA problem). By pigeonhole principle, at least one choice $\{y \in \mathcal{Y}\}$ will have at least $\frac{1}{|\mathcal{Y}|}$ representation among the experts' advice. With each mistake made, the size of the version space (the pool of consistent experts) reduces by at least $\frac{1}{|\mathcal{Y}|}$; that is, $|\mathbf{V}^{(t+1)}| \leq \frac{|\mathbf{V}^{(t)}|}{|\mathcal{Y}|}$.[3] The Halving Algorithm pseudocode is listed in algorithm 1.

Algorithm 1 Halving Algorithm

$\mathbf{V}^{(1)} = \mathcal{H}$ ▷ Version space initially stores all hypotheses
for $t = 1, \cdots, T$ **do**
  RECEIVE $(\mathbf{x}^{(t)})$ ▷ Receive expert prediction
  $h = \text{MAJORITY}(\mathbf{V}^{(t)})$ ▷ Select majority hypothesis
  $\hat{y}^{(t)} = h(\mathbf{x}^{(t)})$ ▷ Rredict using selected hypothesis
  RECEIVE $(y^{(t)})$ ▷ Receive true outcome
  $\mathbf{V}^{(t+1)} \leftarrow \{h \in \mathbf{V}^{(t)} : h(\mathbf{x})^{(t)} = y^{(t)}\}$ ▷ Update Version space, removing inconsistent experts
end for

4.1 Worst-case Mistake Bounds for the Halving Algorithm

Theorem 1. ***(Mistake bound of Halving Algorithm)*** *Let $M_H^{(t)}(\mathcal{H})$ be the total number of mistakes made by the Halving Algorithm by timestep t for hypothesis class $\mathcal{H}$. Assuming realizability and that there is at least one expert in the starting Version space $\mathbf{V}$, $M_H^{(T)}(\mathcal{H})$ is upper-bounded as:*

$$M_H^{(T)}(\mathcal{H}) \leq \log_{\frac{1}{|\mathcal{Y}|}} |\mathcal{H}|$$

Proof. We use induction to establish an upper bound on the size of the Version space at time t. Initially, there are $|\mathcal{H}|$ experts, so our base case is:

$$|\mathbf{V}^{(1)}| = |\mathcal{H}|$$

Now our inductive step. If the algorithm makes a mistake at time t, then:

$$|\mathbf{V}^{(t+1)}| \leq \frac{1}{|\mathcal{Y}|} \cdot |\mathbf{V}^{(t)}|$$

Because the algorithm picks the highest-represented decision at each time t, we know at least $\frac{1}{|\mathcal{Y}|} \cdot |\mathbf{V}^{(t)}|$ experts followed the (wrong) decision we had just picked, and thus at most $\frac{1}{|\mathcal{Y}|} \cdot |\mathbf{V}^{(t)}|$ experts followed the correct decision. Eliminating the $\frac{|\mathcal{Y}|-1}{|\mathcal{Y}|}|\mathbf{V}|$ incorrect experts from the Version space, we are left with just the $\frac{1}{|\mathcal{Y}} \cdot |\mathbf{V}^{(t)}|$ correct experts by timestep $t+1$. So after $M_H^{(t)}$ mistakes, at step t:

$$|\mathbf{V}^{(t)}| \leq \left(\frac{1}{|\mathcal{Y}|}\right)^{M_H^{(t)}} \cdot |\mathcal{H}|$$

This is an upper bound on the size of the version space at time t.
For the lower bound: since we assume realizability, then there is at least one expert at any time t:

$$|\mathbf{V}^{(t)}| \geq 1$$

Putting the upper and lower bounds together, we get:

$$1 \leq \left(\frac{1}{|\mathcal{Y}|}\right)^{M_H^{(t)}} \cdot |\mathcal{H}|$$

Note $0 \leq \frac{1}{|\mathcal{Y}|} < 1$. Simplifying:

$$|\mathcal{Y}|^{M_H^{(t)}} \leq |\mathcal{H}|$$

$$M_H^{(t)} \leq \log_{|\mathcal{Y}|} |\mathcal{H}|$$

□

5 Randomized Greedy Algorithm

The randomized greedy algorithm takes another step toward better mistake bounds, thwarting the adversary by picking hypotheses through uniform sampling. At each time step, the algorithm chooses a random expert that has not yet made a mistake by random and follows its advice. The algorithm is identical to the greedy algorithm, save that the expert is chosen at random from the Version space, as opposed to simply picking the first expert in the Version space. This minor change leads to a significant improvement in mistake bounds in adversarial analysis. [3] The Randomized Greedy Algorithm pseudocode is listed in algorithm 2.

Algorithm 2 Randomized Greedy Algorithm

$\mathbf{V^{(1)}} = \mathcal{H}$ ▷ Version space initially stores all hypotheses
for $t = 1, \cdots, T$ **do**
  RECEIVE $(\mathbf{x}^{(t)})$ ▷ Receive expert prediction
  $h = \text{RANDOM}(\mathbf{V}^{(t)})$ ▷ Random selection from Version space
  $\hat{y}^{(t)} = h(\mathbf{x}^{(t)})$ ▷ predict using selected expert
  RECEIVE $(y^{(t)})$ ▷ Receive true outcome
  $\mathbf{V}^{(t+1)} \leftarrow \{h \in \mathbf{V}^{(t)} : h(\mathbf{x})^{(t)} = y^{(t)}\}$ ▷ Update Version space, removing inconsistent experts
end for

5.1 Worst-case Mistake Bounds for the Randomized Greedy Algorithm

Theorem 2. ***(Mistake bound of Randomized Greedy Algorithm)*** *Let $M_{RG}^{(t)}(\mathcal{H})$ be the total number of mistakes made by the Randomized Greedy Algorithm by timestep t for hypothesis class $\mathcal{H}$. Assuming realizability and that there is at least one expert in the starting Version space $\mathbf{V}^{(1)}$, $M_{RG}^{(T)}(\mathcal{H})$ is upper-bounded as:*

$$M_{RG}^{(t)}(\mathcal{H}) \leq \ln |\mathcal{H}|$$

Proof. We use induction to establish an upper bound on the size of the Version space at time t. Initially, there are $|\mathcal{H}|$ experts, so our base case is:

$$|\mathbf{V}^{(1)}| = |\mathcal{H}|$$

Now our inductive step. If the algorithm makes a mistake at time t, then:

$$|\mathbf{V}^{(t+1)}| = \alpha^{(t)} |\mathbf{V}^{(t)}|$$

where $\alpha^{(t)}$ is the factor our Version space decreases by each time step due to eliminating inconsistent experts – in other words, the "probability" any one expert will be correct on timestep T. Since our algorithm is randomized, even in the worst case α will vary from timestep to timestep. After T timesteps,

$$|\mathbf{V}^{T}| = |\mathcal{H}| \cdot \prod_{t=1}^{T} \alpha^{(t)}$$

We use the useful mathematical inequality $x \leq e^{-(x-1)}$ for real x and simplifying:

$$|\mathbf{V}^{T}| \leq |\mathcal{H}| \cdot \prod_{t=1}^{T} e^{-(1-\alpha^{(t)})}$$

$$|\mathbf{V}^{T}| \leq |\mathcal{H}| \cdot e^{-\sum_{t=1}^{T}(1-\alpha^{(t)})}$$

Note that $1 - \alpha^{(t)}$ is proportion of experts who made a mistake at timestep t. thus, $\sum_{t=1}^{T}(1-\alpha^{(t)})$ is the total number of mistakes that have been made from $t = 1 \cdots T$. That is, $M_{RG}^{(T)}$, the metric we want to bound.

For the lower bound: since we assume realizability, then there is at least one expert at any time t:

$$|\mathbf{V}^{(t)}| \geq 1$$

Putting the upper and lower bounds together, we get:

$$1 \leq \mathcal{H} \cdot e^{-M_{RG}^{(T)}}$$

Rearranging and simplifying:

$$M_{RG}^{(t)} \leq \ln |\mathcal{H}|$$

□

6 Regret

Up till now, our algorithms and analysis are all carried out under the umbrella of assuming realizability–namely, there exists a perfect hypothesis that makes no mistake. Thus, so far we have evaluated our algorithms using the mistake bounds, which tell us how far away our online algorithm is from a perfect hypothesis that makes 0 mistakes. However, if there is no such perfect hypothesis, the mistake bound is no longer a valid evaluation metric (because even the best hypothesis will make mistakes). To measure the performance gap of the "best" (maybe not perfect) hypothesis and our online learning algorithm, we introduce regret.

Formerly, in a hypothesis class $\mathcal{H}$, if we relax the realizability assumption (no longer assume the perfect hypothesis $h^\star$ exist), we want to measure the difference between our online algorithm ℓ and any hypothesis in $h \in \mathcal{H}$. The regret $R^T(h)$ over time period T between ℓ and h is defined as the difference between the cumulative loss of ℓ and h :

$$R^{(T)}(h) = \sum_{t=1}^{T} \ell\left(\hat{y}^{(t)}, y^{(t)}\right) - \sum_{t=1}^{T} \ell\left(h\left(\boldsymbol{x}^{(t)}\right), y^{(t)}\right)$$

The regret between our learner and the best hypothesis (that achieves the cumulative minimal loss) is defined as follows:

$$R^{(T)}(\mathcal{H}) = \max_h R^{(T)}(h) = \sum_{t=1}^{T} \ell\left(\hat{y}^{(t)}, y^{(t)}\right) - \min_h \sum_{t=1}^{T} \ell\left(h\left(\boldsymbol{x}^{(t)}\right), y^{(t)}\right)$$

In online learning, we would like to find an algorithm whose average regret is bounded as T goes to infinity:

$$\begin{aligned}\lim_{T\to\infty} \frac{1}{T} R^{(T)}(\mathcal{H}) &= \lim_{T\to\infty} \sum_{t=1}^{T} \ell^{(t)}\left(\hat{y}^{(t)}, y^{(t)}\right) - \min_{h\in\mathcal{H}} \ell^{(t)}\left(h\left(x^{(t)}\right), y^{(t)}\right) \\ &= 0\end{aligned}$$

or

$$\frac{R^{(T)}(\mathcal{H})}{T} \to 0 \text{ as } T \to \infty$$

If your average regret goes to 0 as T goes to infinity, you have a "no regret " algorithm. For the average regret to be bounded, the regret must grow sub-linearly with respect to T. In the long run, a no regret algorithm performs no worse than the best hypothesis in hindsight.

7 Conclusion

In this lecture, we introduced two more algorithms to solve the PWEA problem, the Halving Algorithm and the Randomized Greedy Algorithm, both of which assumed the realizability. We derived the mistake bound for both of these algorithms, and both showed great improvement over the Consistent Alogrithm.

We also formally introduced the online learning problem, and the setting online learning usually perform in – assuming no realizability. Once we remove the realizability assumption, we can no longer use the mistake bound as our measure for comparing between the best hypothesis and our algorithm. Thus, we introduced regret as an alternative evaluation metric to compare between our learner and the best hypothesis. Assuming no realizability also means that we can no longer remove a hypothesis as soon as it makes a mistake (something all our aforementioned algorithms do). In the next lecture, we will introduce new algorithms that can operate under no realizability assumption, and we will analyze these algorithms using the regret bound.

References

[1] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, May 1997.

[2] N. Cesa-Bianchi and G. Lugosi. On prediction of individual sequences. *The Annals of Statistics*, 27(6):1865–1895, 1999.

[3] K. Kitani. 16-831 lecture slides.